HOW TO GET A TOP GRADE IN YOUR PROJECT

ANALYSIS [10]

What to Include:

- ***How is the problem solvable by computational methods?***
    - Why is a computer program needed?
    - *e.g. requires calculations/accuracy*
    - *e.g. uses large amounts of data*
    - *e.g. needs 3D simulations that can be explored*
- ***Description of the end user(s) / stakeholder(s)***
    - their needs
    - how the system will meet their needs
- ***Investigating existing (or similar) solutions***
    - How do they work?
    - What components/features/approaches will be 'borrowed' and why?
- ***Essential features***
    - With justification
- ***Limitations of proposed solution***
    - What it may not be able to do
- ***Hardware and software requirements***
- ***Measurable success criteria***

Top Band Marks:

- Discuss why your project will be solvable using computational methods
- Clearly identify your stakeholders, and the roles they will play
- Show evidence of looking at other systems/ideas
- Link this research into your proposed designs
- Focus on key features (e.g. GUI Layout, Data structures) and explain and justify why you have chosen these features
- Clearly delimit your problem and identify where you feel your project may face issues/limitations, discussing why these exist and the potential impact on the project

- Generate an overarching set of requirements for the project and also identify hardware/software requirements where needed:
    - This may link to Software Versions (e.g. Only working on certain releases of web browsers due to functionality requirements)
    - Hardware requirements – mostly focusing on unique hardware issues where present. Generic discussion of a minimal set of requirements for a PC is not needed.
- Define a clear set of Success Criteria that is a summary of the above and where each one is *justified* in relation to the Stakeholders /research etc.

DESIGN [15]

What to Include:

- ***Decomposition*** (e.g. structure diagram)
- ***Algorithms*** (flowcharts/pseudocode)
    - Explain how these form the solution (e.g. link to structure diagram)
- ***Usability features***
- ***Describe key structures to be used*** (e.g. comment pseudocode/annotate flowcharts)
    - Variables
    - Data structures
    - Classes
- ***Test data***
    - Does *not* need to be full testing tables

Top Band Marks:

- You should use a suitable design methodology for your chosen project & language. For instance, use of class diagrams for OOP, or Top-Down Design for Procedural.
- Each smaller problem should be justified as to why this is a suitable "chunk" and how it fits in to the grand scheme
- Flow Charts, Data Flow, Pseudocode etc. should be used appropriately to then design the program
- There is no specified design methodology
- Pseudocode is inherently required

- The emphasis is that a 3rd party should be able to implement the solution directly from the designs
- Suitable data structure design should be used as well
- Test data for the initial designs should be specified on a 'module by module' approach, as well as test data that will be used to ensure **all** the Success Criteria are met
- Awarding of marks for Design and Test plans may be awarded at a later date if you decide that your initial designs/tests may need modification/adaptation later in the project

DEVELOPMENT [15]

What to Include:

- ***Show how the solution was created***
    - Write a section of code
    - Show it
    - Test it (next section)
    - Comment on result (show change/correction and further test(s) if needed)
- ***Evidence of prototypes*** *(where used)*
- ***Comment code***
- ***Appropriate identifiers***
- ***Validation***

Top Band Marks:

- Avoid 'death by screen shot'
- Prints of Code blocks/modules with descriptions are fine – no need for 'line by line account'
- Highlight unique features or 'complex coding' etc. – i.e. Showcase the complexity
- Develop in blocks/chunks as defined in the design
- Test as you go – no need to wait until the end
- Agile/Iterative/RAD style development
- Good annotation in code will reduce quantity of writing
- 'Lean and Mean' – keep the development as concise as possible

- Ensure good naming conventions are used (should be included in designs). Using identifiers such as *Form_1* etc., does not show good maintainability of code.
- Highlight the validation – emphasise robust coding
- Check Points/Milestones after each module – review back to *Success Criteria*
- Signed off development by Stakeholders for each iteration

DEVELOPMENT – TESTING [10]

What to Include:

- ***Do not separate from previous section***
- ***Evidence of testing at each stage***
- ***Show any corrections and explain them***

Top Band Marks:

- Test at the end of each 'module'
- No need to screenshot every test – especially if repetitive
- Highlight failed tests
- Must be enough to convince moderator of a working solution
- It is fine to re-design at this point – marks can still be awarded for Design at this stage if they reflect a full working solution
- Any modification of test plans may also be used in the same way
- You **must** show re-development and testing where appropriate to gain full marks
- Ensure all white box testing (functionality) is completed at this stage.
- Testing for User Acceptance (black box) is contained as part of the Evaluation

EVALUATION – TESTING [5]

What to Include:

- ***Completed test plans***
- ***Evidence of results***
- ***Test:***
    - Functionality
    - Robustness
    - Usability

Top Band Marks:

- Black Box testing / User Acceptance Testing
- Involve all Stakeholders
- Identify issues raised from testing
- Again, it is fine to discuss solutions to any 'failures' and repeat the design / implementation phases to solve issues
- Must prove that the system meets all the success criteria
- May reference successes from earlier in the project – no need to repeat
- Reference clearly back to evidence through page numbers (and make sure they are correct!)

EVALUATION [15]

What to Include:

- ***Compare to success criteria***
    - With evidence (or cross reference to where the evidence can be found)
- ***If not/partially met, comment on how it could be addressed in the future***
- ***Discuss success of usability features***
    - If not/partially, how can it be addressed in the future
- ***Discuss potential maintenance issues that may occur***
- ***Discuss limitations with the solution***
- ***Describe how the program could be improved to deal with limitations***

Top Band Marks:

- Summary of final Success Criteria
    - Again – cross reference to any earlier evidence to save repetition
- Identify any potential limitations arising from the solution
    - You may go back and edit/customise if time allows
    - Critique if these are in fact future development problems
- Evidence that GUIs and Features are suitable and effective
- Discuss development of features that may be functional to meet success criteria, but could also be improved
- Explain impact of this development on final product
- Discuss how the system will need to be maintained
    - Refer to code maintenance
    - Evolving requirements of Stakeholders
- Refer to evidence, development, research etc. to support all justifications clearly.
- Ensure that sources are robust and wide-ranging